Problema ***Corte de Barras de Ferro***

Entrada: $\langle n, p \rangle$, onde n é um inteiro positivo indicando o tamanho da barra, sendo que cada tamanho i de barra, com $1 \le i \le p$, tem um preço p_i de venda.
Saída: Maior lucro obtido com a venda de pedaços inteiros de uma barra de tamanho n.

$n = 6$

	1	2	3	4	5	6
p	1	3	11	16	19	10
	1	1,5	3,67	4	3,8	1,67

Soluções viáveis:

6 pedaços de tam. 1 → $6 \cdot p_1 = 6$
3 tam. 2 → $3 \cdot p_2 = 9$
1 tam. 6 → $p_6 = 10$
1 tam. 1 + 1 tam. 2 + 1 tam. 3 → $p_1 + p_2 + p_3 = 15$
1 tam. 5 + 1 tam. 1 → $p_5 + p_1 = 20$
1 tam. 4 + 1 tam. 2 → $p_4 + p_2 = 19$

Sol. ótima:

2 tam. 3 → $2 \cdot p_3 = 22$

Como resolver esse problema?

→ Existe um número finito de soluções viáveis
↳ Descreva uma solução $S = (s_1, s_2, \ldots, s_{n-2}, s_{n-1})$ onde $s_i = 1$ se houve corte à distância i do início da barra e $s_i = 0$ c.c.

Barra de tamanho 6:

$S = (0, 1, 0, 0, 1, 0)$

↳ Existem $\le 2^{n-1}$ soluções

→ Então podemos fazer um algoritmo de força bruta:

```
lucro_max = -1
para cada solução S
    se S é viável e p(S) > lucro_max
        lucro_max = p(S)
devolve lucro_max
```

Como resolver esse problema?

→ Podemos criar um algoritmo guloso: precisamos escolher tamanhos de pedaços da barra.
  ↳ Escolha por melhor valor
  ↳ Escolha por melhor razão $\frac{valor}{tam.}$

→ São polinomiais e viáveis

→ São ótimos?

→ Quando cortamos um pedaço de tamanho i, nos resta uma barra de tamanho $n-i$
  ↳ Temos um subproblema!

→ Alguns cuidados:
  ↳ Caso base: qual o menor tamanho de barra que conseguimos resolver diretamente?
  ↳ Se $i = n$, $n-i$ não reduz o tamanho!

Abordagem Recursiva

```
CORTA (n, p)
    se n == 1
        devolve 0
    escolha um valor i entre 1 e n-1
    S = CORTA (n-i, p)
    se S + p[i] > p[n]
        devolve S + p[i]
    devolve p[n]
```

→ Qual valor de i escolher?
  ↳ De maior p_i?
  ↳ De maior p_i/i?

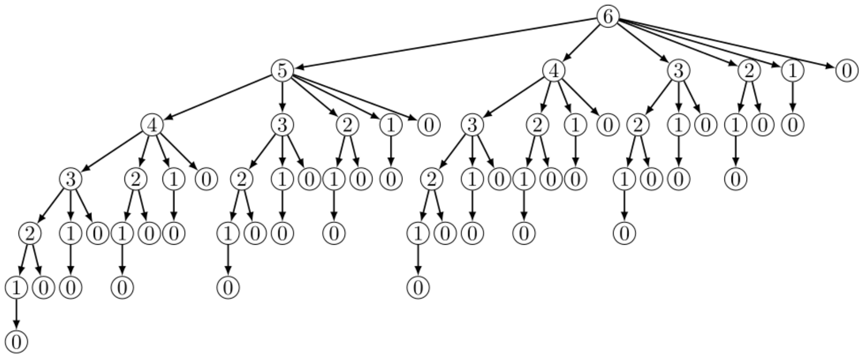
→ São só $n-1$ valores possíveis!

Abordagem recursiva "ótima"

CORTEBARRAS(n)

```
se n == 0 então
    devolve 0
lucro = −1
para i = 1 até n, incrementando faça
    valor = p_i + CORTEBARRAS(n − i)
    se valor > lucro então
        lucro = valor
devolve lucro
```

→ Árvore de recursão para $n = 6$.



Abordagem recursiva

① Esse algoritmo é ótimo?

Se L_t é o lucro ótimo de uma barra de tamanho t, então

$$L_t = \max_{1 \leq i \leq t} \{ p_i + L_{t-i} \}$$

porque a solução ótima para a barra t contém soluções ótimas para barras menores, como provado a seguir.

Seja $S = (c_1, c_2, \dots, c_x)$ uma sequência de cortes ótima para t. Seja c_j um corte qualquer em S. Note que $S - c_j$ deve ser ótima para a barra $t - c_j$. Se não for, seja S' ótima para $t - c_j$. Veja que $S' + c_j$ é viável para t. Porém $S' + c_j$ tem lucro maior do que S (já que S' tem lucro maior que $S - c_j$), uma contradição.

② Qual seu tempo de execução?

```
CorteBarras(n)
se n == 0 então
    devolve 0
lucro = −1
para i = 1 até n, incrementando faça
    valor = p_i + CorteBarras(n − i)
    se valor > lucro então
        lucro = valor
devolve lucro
```

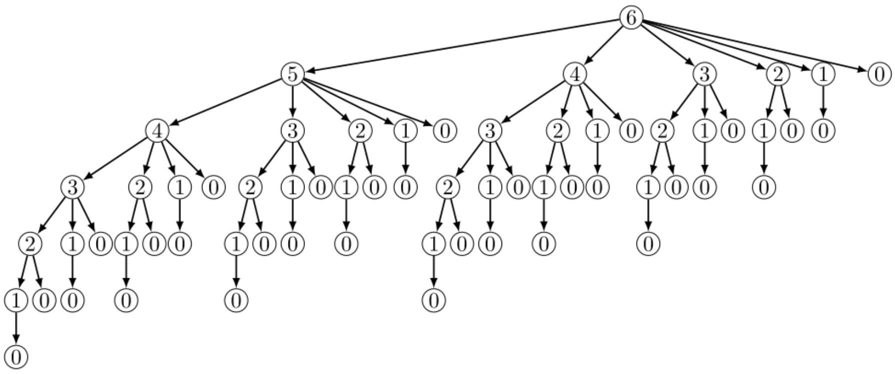
$$T(n) = \sum_{i=1}^{n} T(n-i) + n$$

Pelo método da substituição, $T(n) \geq 2^n$

$$T(n) = n + \sum_{i=1}^{n} T(n-i) \geq n + \sum_{i=1}^{n} 2^{n-i}$$
$$= n + 2^{n-1} + 2^{n-2} + \dots + 2^0 = 2^n - 1 + n$$
$$\geq 2^n \quad (n \geq 1)$$

③ Dá para melhorar?

Note que existem subproblemas repetidos sendo calculados várias vezes.



Mas só existem $n+1$ subproblemas ao todo!

↳ Um para cada tamanho de barra, de 0 a n.

↳ Cada um pode ser descrito por um valor i, $0 \leq i \leq n$

↳ Um vetor $B[0..n]$ consegue armazená-los.

↳ $B[i]$ terá o lucro máximo obtido ao cortar uma barra de tamanho i

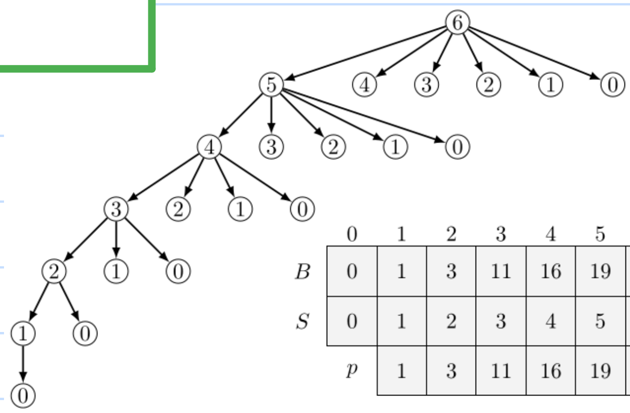
↳ Queremos calcular $B[n]$

PD para Corte de Barras (abordagem top-down)

```
CorteBarrasRecursivo-TopDown(k)
se B[k] == −1 então
    lucro = −1
    para i = 1 até k, incrementando faça
        valor = p_i + CorteBarrasRecursivo-TopDown(k − i)
        se valor > lucro então
            lucro = valor
            S[k] = i
    B[k] = lucro
devolve B[k]
```

	1	2	3	4	5	6
B	-1	-1	-1	-1	-1	-1
S						

```
CorteBarras-TopDown(n)
Cria vetores B[0..n] e S[0..n] globais
B[0] = 0
para i = 1 até n, incrementando faça
    B[i] = −1
devolve CorteBarrasRecursivo-TopDown(n)
```



	0	1	2	3	4	5	6
B	0	1	3	11	16	19	22
S	0	1	2	3	4	5	3
p		1	3	11	16	19	10

PD para Corte de Barras (abordagem bottom-up)

```
CorteBarras-BottomUp(n)
Cria vetores B[0..n] e S[0..n]
B[0] = 0
para k = 1 até n, incrementando faça
    lucro = −1
    para i = 1 até k, incrementando faça
        valor = p_i + B[k − i]
        se valor > lucro então
            lucro = valor
            S[k] = i
    B[k] = lucro
devolve B[n]
```

	1	2	3	4	5	6
B	-1	-1	-1	-1	-1	-1
S						

Construindo uma solução

→ Para cortar a barra n e obter lucro ótimo $B[n]$, houve um corte de tamanho $S[n]$.

→ Nos resta uma barra $n - S[n]$, portanto.
Para o lucro ótimo dela, $B[n - S[n]]$, houve um corte de tamanho $S[n - S[n]]$.

```
enquanto n > 0 faça
    Imprime S[n]
    n = n − S[n]
```